John T. Anderson

Engineering Note

Date: October 15, 1999
Rev Date: ~~October 15, 1999~~ ~~**April 19,2000**~~ **September 13, 2001**

Project: CFT
Doc. No: A1000420

Subject: SVX and Virtual SVX operation in the Analog Front End Board

Introduction

A recent deluge of questions regarding the Virtual SVX impels me to write this note describing, in as painful detail as possible, exactly how the VSVX is supposed to work. Of course, since most of the VSVX is implemented in programmable logic, some change is allowable and probably inevitable; none the less, this document describes the design's intent and current implementation.

This document is a major departure from the original document, #A991015. These changes are due to simplifications in the logic caused by allowing the VSVX to obtain its initialization information from the on-board microprocessor rather than the SVX initialization string, and also changes in the FIFO architecture surrounding the VSVX which allow some tasks to be relegated to microprocessor firmware.

This September 13, 2001 document is a relatively minor departure from the April 19, 2000 release. Items in that release not implemented in the production boards are deleted using ~~strikethrough~~ effects. New information is <u>underlined.</u>

<u>This September 13, 2001 document eliminates all reference to the 12-MCM AFE and describes only the Virtual SVX as implemented on the Revision 1C, 8-MCM AFE boards.</u>

System Overview

In the standard Central Fiber Tracker Axial Analog Front End board with eight multi-chip modules (hereafter referred to by the acronym 8-MCM AFE), every input channel is sampled by a discriminator (the SIFT,or **SIF**i **T**rigger[1] chip) and an eight-bit ADC, the SVX IIe (from **S**ilicon **V**erte**X**). The discriminator patterns are analyzed every crossing and those indicative of an interesting event are further analyzed by enabling the digitization and readout of the SVX chips as well. An analog pipeline in the SVX chip holds the charge data for up to 32 crossings of delay, to allow the trigger system time to reach its decision. Only upon acceptance of an event does the SVX actually digitize and convert the charge into values. If a trigger is not issued within 32 crossings of an event, the charge is lost and replaced by charge collected in a new event.

To ensure correct correlation of data, the discriminator patterns are stored in FIFO buffers on the AFE and, should a SVX readout ensue, the discriminator pattern which caused the SVX readout is appended to the SVX data as a 'fake' or 'virtual' SVX chip. Thus, the 8-MCM AFE appears to have nine SVX chips, not eight. To minimize readout delay, the SVX chips on the 12-MCM AFE are split into two readout streams. The first stream has four 'real' and one 'virtual' SVX for a total of five, and the second stream has eight 'real' and one 'virtual' SVX for a total of nine. Optional diagnostic modes allow the Virtual SVX system (VSVX) to include board diagnostic information from the microprocessor; in these modes the VSVX looks like two extra SVX chips in the readout chain, not just one.

SVX Data Format and Initialization

The data provided by a 'real' SVX chip is byte-oriented. Readout of a chip starts when it receives a token (PRIORITY_IN) and a new data value is asserted onto the data bus with each rising edge of the SVX clock. The chip continues to assert data until it is empty, at which point the chip asserts PRIORITY_OUT, which is presumably the PRIORITY_IN of the next chip in sequence. When the entire chain is done the last PRIORITY_OUT goes back to the readout controller, signifying that all data has been sent. Every data word is associated with a transition in the DVALID line.

[1] Acronym definition thanks to Fred Borcherding.

This line is used by the readout controller as the strobe to latch the data. The board receiving the data utilizes a clock doubler to obtain a rising clock edge for each transition in DVALID; the data source must assert new data for each transition of the DVALID line.

A fixed format is utilized by the SVX chip in its data as shown in Table 1. The first byte is a CHIP_ID value which uniquely identifies which chip in the chain this data comes from. The second byte is a STATUS value which indicates some internal chip parameters. Following the CHIP ID/STATUS pair, the data is asserted as a CHANNEL (or ADDRESS) byte followed by a DATA byte. The Virtual SVX may be identified uniquely via the CHIP_ID value(s) it presents. In the D-Zero Fiber Tracker system, no 'real' SVX chip will ever be assigned a CHIP_ID greater than 13 decimal. Historically, the maximum allowed CHIP_ID is 15 decimal.

CHIP_ID value 15 is administratively reserved for use by the Virtual SVX when presenting discriminator information. CHIP_ID value 14 is administratively reserved for use by the Virtual SVX when presenting board status, diagnostic or other non-discriminator information.[2] In the implementation of the production (revision 1C) AFE, chip ID 14 is never used.

Index	Interpretation
1	CHIP_ID
2	STATUS
3	CHANNEL #
4	DATA
5	CHANNEL #
6	DATA
7….n	More channel/data pairs

Table 1

The SVX chip can operate in different readout modes. In READ_ALL mode, every channel of the chip reads out regardless of data value. If the READ_ALL is not set the chip operates in a zero suppression mode where only those channels above a programmed threshold show up. Another mode called READ_NEIGHBOR exists which, during zero suppressed readout, also enables those channels adjacent in channel number to those channels who are above the suppression threshold.

Note that, because the MCM on the AFE uses only some of the channels in the SVX – only 72 of the 128 channels are actually connected, and they're not adjacent – the READ_NEIGHBOR bit is essentially useless in the AFE.

Setup of the SVX chip is done through a serial procedure where 190 bits of configuration data are downloaded to each chip in turn. A group of SVX chips looks like one big shift register of N*190 bits, where 'N' is the number of SVX chips in the chain. The bits are identified in the SVX IIe "Beginner's Guide" available on the Internet at http://d0server1.fnal.gov/projects/silicon/www/svx2e/svxe.html.

The Virtual SVX logic in no way participates in the serial initialization procedure, except that it must connect to the pins used for this download to participate in the readout. This subsystem derives its control values from internal registers which are loaded by the on-board microprocessor in response to commands from the MIL-STD 1553 interface (e.g, the experiment slow monitor system). Although many bits are necessary to initialize the real SVX chip, the Virtual SVX requires little in the way of initialization. By convention the CHIP_ID is fixed, and so need not be downloaded. The VSVX has no analog to digital conversion functions; it only collects and transfers digital data.

Block Diagram of Virtual SVX Implementation on the AFE

Figure 1 shows a sketch of how the Virtual SVX is connected to the other portions of the Analog Front End board. The sketch obviously ignores much of the circuitry on the AFE in order to more clearly show the functioning of the Virtual SVX. All eight analog input blocks feed the Virtual SVX, which manipulates the various FIFOs to create the excess SVX data for the SVX Sequencer to read. The drawing most closely describes the setup of the 8-MCM AFE; ~~in the 12-MCM~~

[2] As per discussion between M. Fortner, J. Anderson and K. Papageorgiou, March 29, 2000.

~~variant the motherboard has fewer MCMs to work with, but the daughterboard is much the same as the setup of the 8-MCM board. The 12-MCM design uses two separate SVX readout chains, one for the motherboard, one for the daughterboard.~~ As of this writing, the 12-MCM AFE project has been discontinued.

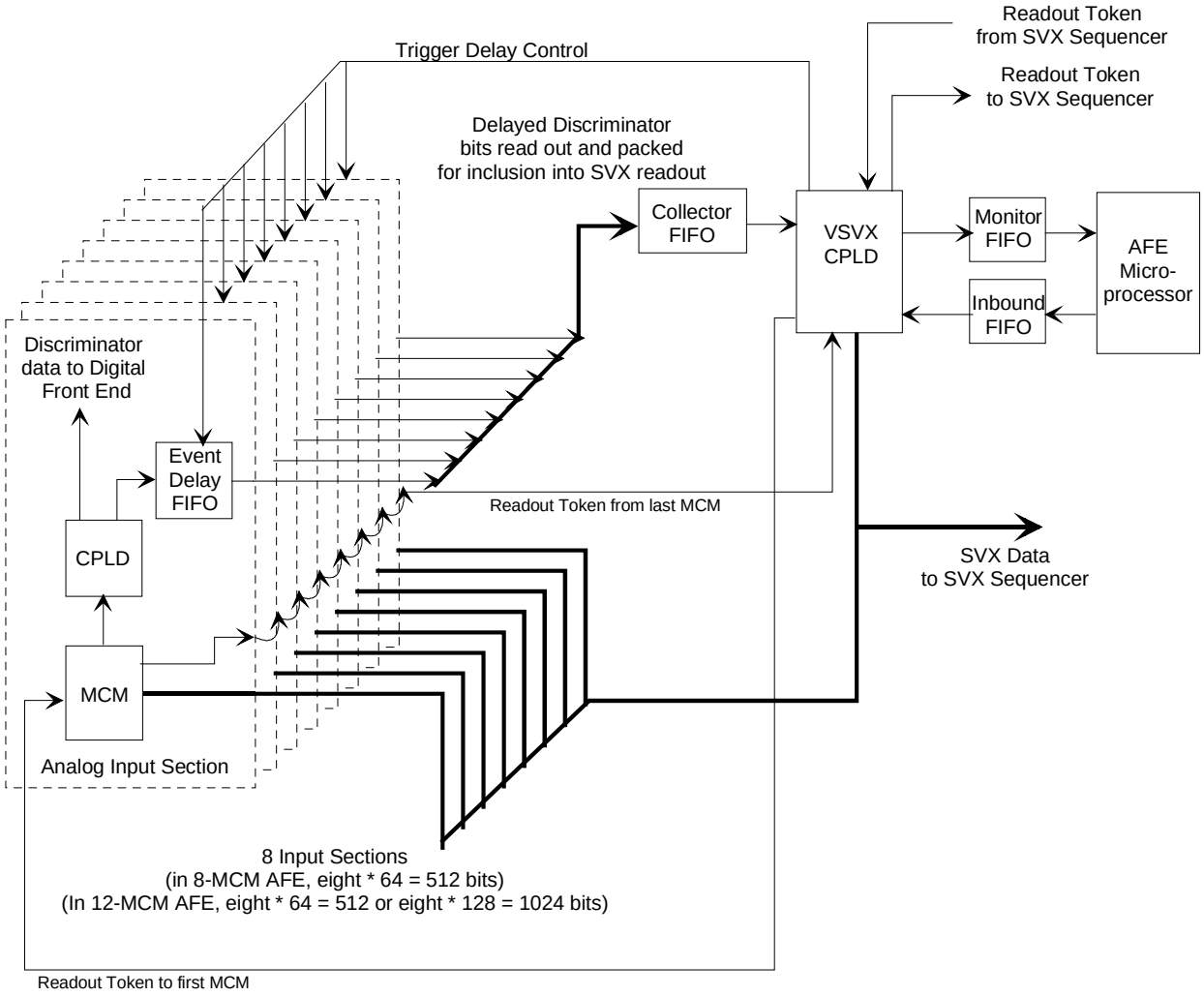

Figure 1

The Virtual SVX has the basic function of capturing the discriminator data from the MCMs and delaying it in the FIFOs in each Analog Input Section such that the data available at the output of each of those FIFOs is synchronized to the pipeline delay in the real SVX chips in the MCMs themselves. The trigger decision typically occurs some 3.8 usec (29 crossings)[3] after the event actually occurs. The SVX has a 32-crossing-deep analog pipeline, and so the decision must occur before the analog data falls off the end. To match the SVX analog pipeline depth, each Analog Input Section ~~implements~~ need implement a FIFO which can store ~~72~~ 64 bits * 32 crossings = ~~288~~ 256 bytes of data. ~~Actually, in the 8-MCM AFE, no MCM has more than 64 active bits, so a 256-element FIFO may be used to save a little cash.~~ Fortunately, the designer of the AFE knows that one should never put an exact fit in, because no specification is ever held sacred at Fermilab. Thus, the AFE is produced with 512 byte FIFOs, allowing a VSVX pipeline depth up to 64 crossings.

The CPLD (**C**omplex **P**rogrammable **L**ogic **D**evice) associated with each MCM (the VSVX_MUX) has the job of determining when the SVX chip has been reset and controlling the event delay FIFOs appropriately. The event delay FIFOs are reset at board power-up and whenever the SVX is read out. The ~~CPLD~~ VSVX_MUX monitors the system for these two

[3] The actual number of crossings is likely to be larger than this. However, if the number of crossings required to form the trigger increases past 31, then all the timing will have to change because the SVX pipeline will overflow at the 132 nsec/tick rate. The probable methodology would be to use a 396 nsec SVX clock. Changing the SVX Acquire clock to a 396 nsec period rather than a 132 nsec period will have no effect upon the VSVX.

conditions and, following each, for when the SVX chip is placed back into the acquire mode. The event delay FIFOs are re-enabled when the SVX goes back into acquire mode such that the FIFOs are buffering at the same time that the internal analog pipeline of the SVX is buffering. A counter within the VSVX CPLD counts the number of crossings after the FIFO has been re-enabled. When this count reaches the count equivalent to the pipeline depth that was programmed into the SVX chips during initialization, the event delay FIFO read clock is started, so that the data at the output pins of the FIFOs always corresponds to the event which will be read out of the SVX should a trigger occur.

Should a trigger erroneously occur before the SVX pipeline depth is reached, the SVX chips won't have data but the Virtual SVX will. The trigger should avoid this condition, but if it doesn't, the Virtual SVX will not emit the correct data. The state machine logic of the VSVX follows the SVX state as controlled by the sequencer and it always assumes that sufficient delay has elapsed within the Acquire mode for the Event Delay FIFOs to fill sufficiently before any transition out of the Acquire mode. The general response of the VSVX to a too-early trigger would be to read out the first event stored in the Event Delay FIFOs since the last reset or readout.

~~Two other FIFOs, the Monitor and the Inbound, provide two-way communication between the CPLD and the AFE board's local microprocessor. In normal operation the Inbound FIFO will contain board status and event environment information loaded by the microprocessor, such as reference voltage settings, operational mode (e.g. 132 ns vs. 396 ns) or run ID data previously stored into the AFE from the 1553 bus. The Virtual SVX may append this status data to the discriminator data in order to form a more complete event record. However, since most of this information should already be available to the analysis through the slow monitoring, the user will have the option to minimize the readout time by skipping this data.~~

While physically implemented in the production AFE, the current firmware load in the VSVX and in the AFE's microprocessor do not support the use of the Monitor and Inbound FIFOs. These can be enabled with different firmware loads on the test bench but CAN NOT BE ENABLED REMOTELY.

~~Alternatively, the Monitor FIFO may be used by the Virtual SVX to store the data from the discriminators, for readout by the microprocessor. This mode is intended to provide a diagnostic path to allow the bench test software access to the SIFT data without requiring the LVDS links to the Digital Front End board. Since the SVX data bus is bidirectional as well, this functionality also allows a slow diagnostic path by which the SVX data may be monitored, possibly useful to detect bit errors in transmission to the SVX Sequencer.~~

Setup and Control of the Virtual SVX

The Virtual SVX is set up through data values written to internal registers via the AFE's on-board microprocessor. For token passing purposes the VSVX has TopNeighbor and BottomNeighbor pins, but the VSVX does not respond to any serial data in the Initialize mode. When the SVX string is being initialized, the VSVX is completely invisible to the SVX Sequencer. The VSVX only becomes visible to the SVX Sequencer during the Readout mode. *Note, however, that one of the functions of the Virtual SVX does change the physical layout of the token passing chain and, if incorrectly set, may disable the initialization of the SVX chips inside the MCMs.*

Internal Registers Implemented in the Virtual SVX

The Virtual SVX implements three internal registers which are programmed through the on-board microprocessor and the MIL-STD 1553 interface.

Virtual SVX Control Register 1

Control Register 1 dictates the direction of data flow and the type of data presented by the Virtual SVX. It is a bitmap control register as detailed in Table 4. Data is loaded into this register by the on-board microprocessor via the 8-bit bidirectional data bus and the WRT_VSVX_REG1 decoded control line. The register is write-only.

A few words are probably required to explain these bits. During shakedown of the system it will be useful to have a known source of consistent data to insure that the SVX readout is reliable. This is accomplished ~~by using data loaded by the microprocessor into the Inbound FIFO and, during readout cycles, emitting only that data.~~by using the VSVX, and only the VSVX, whose data length and pattern are completely predictable. This is accomplished by setting the SKIP_OTHER_CHIPS bit. In this configuration the VSVX will accept the readout token from the SVX Sequencer and respond immediately to it, but when done with readout sending the token back directly to the Sequencer, blocking the token to the 'real' SVX chips. Thus, only the VSVX will appear in the readout chain. ~~In addition, clearing the ENABLE_SIFT bit and setting the ENABLE_STAT bit will disable any readout of SIFT discriminator data, only sending the data from the Inbound FIFO. Any arbitrary test pattern may then be loaded into the Inbound FIFO~~ Further, the VSVX_MUX parts may be set to 'test mode' in which they replace the MCM data by data from a counter.. ***If the SKIP_OTHER_CHIPS bit is set, the***

token from the Sequencer will not be 'seen' by any of the SVX chips in the MCMs themselves. Setting this bit will cause errors in the SVX Initialization function and will prevent proper initialization and/or reset of the SVX chips. Do not use this bit except under well-controlled, bench-test diagnostic conditions.

Once the basic functionality of the SVX readout is established, occasional redundant readout may be required to check for bit errors, without interfering with the normal operation of the board. This is accomplished through the Monitor FIFO. The Monitor FIFO can be filled with either the data 'seen' at the output of the Collector FIFO (i.e., discriminator data) or with data asserted onto the SVX bus by devices other than the Virtual SVX (i.e., a 'spy' upon the data transmitted by the MCMs). The MONITOR_SELECT bit chooses one of these two data sources. The MONITOR_ENABLE bit is set by the microprocessor asynchronously to SVX operation. If MONITOR_SELECT is 0, the Monitor FIFO samples the disciminator data and the MONITOR_ENABLE will clear itself at the end of the next full Digitize cycle. If MONITOR_SELECT is 1, the Monitor FIFO samples the SVX bus and the MONITOR_ENABLE will clear itself at the end of the next full Readout cycle. It should be noted that the microprocessor firmware has little to no support for this feature, intended for use ONLY at the test bench.

Once the AFE is installed in the system, the amount and type of data presented to the SVX Sequencer is controlled using the ENABLE_SIFT ~~and ENABLE_STAT~~ bits. Setting ENABLE_SIFT causes the Virtual SVX to show up in the readout as CHIP_ID 15, with data bytes that contain the SIFT discriminator bits associated with the SVX data readout from the rest of the board. Time correlation of the discriminator information with the SVX readout is accomplished through a pipeline depth value stored in a different register. ~~Setting ENABLE_STAT causes the Virtual SVX to show up in the readout as CHIP_ID 14, with data bytes copied from the Inbound FIFO to the SVX readout. In this scenario the Inbound FIFO is assumed to contain board status and/or environmental data. Since the microprocessor runs asynchronous to the VSVX and, presumably, slower, the IBFIFO_READY bit is used to further qualify the ENABLE_STAT readout. The readout of the Inbound FIFO is destructive and so, if ENABLE_STAT is set, the Inbound FIFO will only be read if the microprocessor has also set the IBFIFO_READY bit to indicate that the FIFO has been refilled.~~

bit position	Bit Name	Function if set (1)	Function if clear (0)
0	SVX_STRING_LENGTH[4]	~~Expect four SVX chips in readout after VSVX~~ Originally intended as a flag to differentiate between 8-MCM and 12-MCM AFEs; now simply an extra bit.	~~Expect 8 SVX after VSVX~~
1	~~ENABLE_STAT~~[5]Unused Bit	~~Enables readout of board status data from BIFIFO as part of Virtual SVX data.~~unused.	~~Disables readout of BIFIFO during SVX readout.~~unused
2	ENABLE_SIFT	Enables readout of SIFT discriminator data from Event Delay FIFOs as part of Virtual SVX data.	Disables readout of SIFT discriminator data from Event Delay FIFOs.
3	~~IBFIFO_READY~~Unused Bit	~~Virtual SVX may read from Inbound FIFO if ENABLE_STAT is set; e.g., micro is done filling Inbound FIFO.~~Unused	~~VSVX will not read from Inbound FIFO even if ENABLE_STAT is set.~~Unused
4	MONITOR_SELECT	Virtual SVX will capture all SVX data from other SVX chips in string to ~~BIFIFO~~ Monitor FIFO during next readout, for alternate readout via microprocessor, in addition to normal readout activities.[6]	~~Data from other SVX chips will not be captured.~~Monitor FIFO will capture data asserted by Virtual SVX, but not any of the 'real' SVX data.[7]
5	MONITOR_ENABLE	~~Trigger from microprocessor telling VSVX to capture selected data to monitor at next correct transition of SVX state.~~Monitor FIFO Write Enable is enabled.	~~No effect.~~Monitor FIFO Write Enable is disabled.
6	SKIP_OTHER_CHIPS	The Virtual SVX uses the PRIORITY_IN from the Sequencer as it's PRIORITY_IN, not the PRIORITY_OUT of the previous chip, thus shortening the length of the readout chain to only one chip – itself.	The chain is allowed to act normally such that the Virtual SVX and all other chips show up as expected.
7	~~RESERVED~~MONITOR_RESET	~~Reserved for future use.~~Holds Monitor FIFO in reset.	~~Reserved for future use.~~Releases Monitor FIFO reset line.

Table 2

[4] This bit is set by the micro in response to a 1553 command and is provided in the VSVX Status word read back on the SVX bus to indicate data set size to packing software.

[5] If software fails to set ~~either the ENABLE_STAT or the~~ ENABLE_SIFT bits, the Virtual SVX will only issue a CHIP_ID and a STATUS word, and no other data.

[6] Note that, because the Monitor FIFO only samples on the rising edge of the SVX clock, only Chip ID and Address data will be captured from the SVX chips. Status and Data values are sent on the falling edge of the SVX clock.

[7] When the Monitor is capturing the VSVX data, only the DATA portion, not the addresses, are stored.

Virtual SVX Control Register 2

VSVX control register 2 is reserved for engineering diagnostics.

Virtual SVX Control Register 3

VSVX control register 3 stores a delay count which is used to set the digital delay of the Event Delay FIFOs. The Virtual SVX state machine operates at ~~69~~ 61 MHz (~~nine~~ eight times the crossing frequency), which is the same speed that the Event Delay FIFOs are filled. Each Event Delay FIFO stores ~~nine~~ eight bytes (64~~72~~ bits) of data every crossing, which is the basis of this clock frequency. The AFE's clock generator uses a phase-locked loop to synthesize the frequency from the crossing clock. In typical operation the VSVX state machine needs to wait 29 crossings, or 261 clock ticks, after the Acquire mode is set before enabling the read clock of the Event Delay FIFOs. This delay is used to allow the Event Delay FIFOs to buffer sufficient information such that, as the acquisition continues, the data available at the output of the FIFO is the data from 29 crossings ago, emulating the pipeline delay of the SVX chip itself. However, trigger timing or other delays may require some adjustment of this value. The VSVX state machine provides a fixed delay of 64 clocks, plus the delay programmed into Control Register 3, such that any delay from 65 to 319 clocks (~~7.22~~8.125 crossings to ~~35.44~~39.875 crossings) may be set. The delay is adjustable in one-clock-tick (1/8~~9~~th crossing) increments.

Production testing has shown that Left-Handed AFE boards are offset by one tick of this count relative to Right-Handed AFEs. Left-Handed boards should be programmed with one LESS than the Right-Handed boards.

Details of Virtual SVX Readout Format

The Virtual SVX follows the same generic readout format as given in Table 1 for the SVX IIe chip. Specific rules are followed:

- The CHIP_ID returned will be 15 for SIFT data, 14 for Inbound FIFO (Status) data.
- The STATUS word will be a bitmap status of the Virtual SVX, taken directly from SVX Control Register 1, as given in Table 5:

Bit #	Interpretation if set (= 1)	Interpretation if clear (= 0)
7	~~Reserved~~Monitor FIFO is held reset	~~Reserved~~Monitor FIFO is active
6	SKIP_OTHER_CHIPS set	SKIP_OTHER_CHIPS clear
5	Monitor FIFO is enabled	Monitor FIFO is disabled
4	Monitor data, if available, is SVX data	Monitor data, if available, is discriminator data
3	~~Length of Status data, if Inbound FIFO data enabled, is non-zero; FIFO was ready~~Simple readback of bit value; no interpretation.	~~Length of Status data, if Inbound FIFO data enabled, is zero; FIFO was not ready~~Simple readback of bit in register; no interpretation
2	Discriminator data enabled	Discriminator data not enabled
1	Simple readback of bit value; no interpretation.~~Inbound FIFO data enabled~~	Simple readback of bit in register; no interpretation~~Inbound FIFO data not enabled~~
0	Simple readback of bit in register; no interpretation~~SVX Chain Length = 4 chips~~	Simple readback of bit in register; no interpretation~~SVX Chain Length = 8 chips~~

Table 3

- The ADDRESS portion of the two-byte address/data pair will indicate the source of the data (i.e., which MCM and which byte from that MCM) according to Table 6. The general algorithm used is that data from the Event Delay FIFOs will have an ADDRESS field which is, in binary, 0**mmmdddd** where the **mmm** is a three bit value indicative of which

MCM on the board sourced this data (MCM #0- #7) and the **dddd** field indicates which byte number (0-~~8~~7) of the bytes of data from that MCM this comes from.

Each MCM may possibly source as many as 72 bits of discriminator data, which would require 9 bytes of data per MCM. The ADDRESS field algorithm above thus reserves 72 of the possible 128 addresses for MCM data in eight blocks, leaving 56 addresses in eight other blocks unused. Although nothing physically prevents the Virtual SVX from using addresses 128 through 255, the SVX convention is that the address byte never has the most significant bit set. Some readout software may use this for validity checking and so the Virtual SVX conforms to this convention.

Address Range	**Data Source**
0x00 – 0x07~~8~~	MCM #0, data bytes 0 through 7~~8~~.
0x0~~8~~9 – 0x0F	Reserved addresses
0x10 – 0x17~~8~~	MCM #1, data bytes 0 through 7~~8~~.
0x1~~8~~9 – 0x1F	Reserved addresses
0x20 – 0x27~~8~~q	MCM #2, data bytes 0 through 7~~8~~.
0x2~~8~~9 – 0x2F	Reserved addresses
0x30 – 0x37~~8~~	MCM #3, data bytes 0 through 7~~8~~.
0x3~~8~~9 – 0x3F	Reserved addresses
0x40 – 0x47~~8~~	MCM #4, data bytes 0 through 7~~8~~.
0x4~~8~~9 – 0x4F	Reserved addresses
0x50 – 0x57~~8~~	MCM #5, data bytes 0 through 7~~8~~.
0x5~~8~~9 – 0x5F	Reserved addresses
0x60 – 0x67~~8~~	MCM #6, data bytes 0 through 7~~8~~.
0x6~~8~~9 – 0x6F	Reserved addresses
0x70 – 0x77~~8~~	MCM #7, data bytes 0 through 7~~8~~.
0x7~~8~~9 – 0x7F	Reserved addresses

Table 4

- Each bit of the DATA field corresponds to one discriminator, and thus, to one SVX channel. Other CPLDs will actually pre-format the data into the Event Delay FIFOs, and the order in which the bits correlate to SVX channels are controlled by their programming. As of this writing no formal order has been agreed upon, but analysis of the resources within those CPLDs indicate that a simple pattern is available.

  The reader is warned that the SVX channel numbers used in the MCM are not contiguous. The wirebond pattern in every MCM may also not be the same due to manufacturing difficulties. Table 7 shows a *typical* association, but software must be capable of providing a *unique association per MCM.*

Byte	**Bit 7 (MSB)**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0 (LSB)**
0	Channel 13	Channel 12	Channel 11	Channel 10	Channel 7	Channel 6	Channel 5	Channel 2
1	Channel 27	Channel 26	Channel 23	Channel 22	Channel 21	Channel 20	Channel 17	Channel 16
2	Channel 42	Channel 41	Channel 38	Channel 37	Channel 36	Channel 32	Channel 29	Channel 28
3	Channel 56	Channel 53	Channel 52	Channel 49	Channel 48	Channel 47	Channel 46	Channel 43
4	Channel 71	Channel 68	Channel 67	Channel 66	Channel 62	Channel 59	Channel 58	Channel 57

5	Channel 83	Channel 82	Channel 81	Channel 78	Channel 77	Channel 74	Channel 73	Channel 72
6	Channel 98	Channel 97	Channel 94	Channel 92	Channel 91	Channel 88	Channel 87	Channel 84
7	Channel 110	Channel 109	Channel 108	Channel 107	Channel 104	Channel 103	Channel 102	Channel 99
8	Channel 124	Channel 123	Channel 120	Channel 119	Channel 118	Channel 117	Channel 114	Channel 113

Table 5

As an example, if the Virtual SVX spits out address 0x63 with the data value 0xA8, this means that SVX channels 56,52 and 48 of MCM #6 had their corresponding discriminator bits set.

~~*Inbound FIFO Status Data*~~

~~The format of Inbound FIFO data is entirely controlled by what the microprocessor loads into the Inbound FIFO for readout. As microprocessor firmware for the AFE is finalized, a separate note describing the Inbound FIFO data will be issued. However, general observations are available now:~~

- ~~Inbound FIFO data will be preceded by a CHIP ID byte whose value is determined by the secondary Chip ID value loaded in the Initialization string.~~
- ~~INBOUND FIFO data will be preceded by the STATUS byte whose value will be zero, except in the collision condition where an SVX readout occurs while the microprocessor is updating the INBOUND FIFO information. In that case, the STATUS byte shall be nonzero and indicate what portion, if any, of the INBOUND FIFO data is valid.~~
- ~~There will never be more than 256 bytes of INBOUND FIFO data, and usually far fewer than that.~~
- ~~In general, INBOUND FIFO data will *not* follow the ADDRESS/DATA pair convention, but shall be all DATA.~~
- ~~The length of the INBOUND FIFO data is variable, but will change very infrequently. Changes in the length of the INBOUND FIFO data can be synchronized to major startup/initialization sequences, such that INBOUND FIFO data length should be consistent throughout a given run.~~

The functionality of the Inbound FIFO is disabled in the production AFE.

Deadtime Analysis

Readout of the SVX data in the CFT system is a significant contributor to experiment deadtime. To insure that the Virtual SVX doesn't cause a big problem here, the deadtime contribution of the Virtual SVX in various configurations need be explored.

'Normal' mode, SIFT Data only

In this mode, the Virtual SVX simply grabs the data from the Event Delay FIFOs and spits it out unprocessed to the SVX Sequencer during readout. The Virtual SVX learns that the event has been accepted by either sensing the L1_ACCEPT signal from the SVX Sequencer or by seeing the SVX mode change from Acquire to Digitize. Either way, the Virtual SVX will have a leadtime equal to the time it takes for the SVX chips to digitize; however, the VSVX has to be ready to be the first to readout, so none of the SVX chip readout time is available for slop.

A reasonable estimate for this lead time is to assume that the SVX will be allowed half of the maximum number of counts for digitization, plus about 1 usec for FIFO collapse. Thus, the conservatively estimated lead time is

$$T_{lead} = (127 * 18.9ns) + 1000ns = 3400ns$$

If one assumes that the Virtual SVX runs at 69 MHz and it takes two clock ticks to fetch each byte from the Event Delay FIFOs, this allows the Virtual SVX to prefetch approximately 117 bytes before it can ever see the token to start readout. The densest string of Event Delay information is found in the 8-MCM string of the 12-MCM AFE, where the Virtual SVX must read out 9 bytes from each of 8 MCMs, or 72 bytes. In the 8-MCM AFE, the clock rate of the entire Event Delay and VSVX system may be relaxed to 61 MHz (eight ticks per crossing, rather than nine), and the same time calculations hold. Thus, it would appear that the Virtual SVX will always be ready before it can get the token, and does not contribute anything more than its data readout time to the deadtime. Assuming eight MCMs, each of which has 9 bytes worth of data, plus a CHIP_ID and a Status word, the longest readout time for the VSVX may be expected to be

$$T_{read} = (9*8*2*18.9ns) + (2*18.9ns) = 2.76u\,\text{sec}$$

Effects of ENABLE_STAT and/or Monitor FIFO usage

Turning on the ENABLE_STAT bit increases the readout deadtime because there is more data to be read out. The amount of time will be controlled by how much data is put into the Inbound FIFO by the microprocessor. A fixed penalty of

Error Handling

If the SVX Sequencer issues a trigger too quickly after putting the SVX chain into Acquire mode, obviously both the VSVX and the 'real' SVX chips will be unable to deliver the desired event. Since this error must be handled by the Sequencer, the VSVX uses a simple protocol of not being able to accept the change until after the requisite pipeline delay has elapsed.

Any other errors are simply handed by resetting the state machines. The state machines reset upon power up, but enough excess resources exist to emplace a 'soft' reset via the on-board microprocessor.